Azure AI Agent Delivery System Architecture

Architecture Overview

This architecture illustrates how AI agents can be delivered within an existing application using Azure services. The system supports streaming responses, agent configuration, authenticated actions, and evaluation frameworks.

Core Azure Services | Data & Storage | Security Components | Integration & Developer Tools

Client Layer

Client Application
Chat interface with token-by-token streaming support via SignalR

API Gateway Layer

Azure API Management
Manages APIs, handles throttling, caching, and authentication delegation

Application Layer

Azure App Service
Hosts main application backend with WebSocket support

Azure SignalR Service
Manages real-time WebSocket connections for streaming

Azure Container Apps
Hosts isolated agent environments with containerization

AI & Serverless Layer

Azure OpenAI Service
Provides managed access to OpenAI models with streaming support

Azure Functions
Serverless execution of agent tools and background tasks

Data & Storage Layer

Azure Cosmos DB
Stores agent configurations, evaluations, and versioning data

Azure Blob Storage
Stores chat logs, evaluation datasets, and model artifacts

Security Layer

Azure Active Directory
Handles authentication, authorization, and user identity

Azure Key Vault
Secures API keys, credentials, and certificates

Integration & Event Layer

Azure Event Grid
Manages event-driven architecture for async operations

Azure Service Bus
Message queuing for reliable async processing

Monitoring & DevOps Layer

Azure Application Insights
Monitors performance and usage metrics

Azure DevOps
CI/CD pipeline and versioning control

Azure Feature Flags
Controls feature rollout and A/B testing

Key Data Flows

Chat Request Flow

1. User submits message through client application
2. Request is authenticated via Azure AD and goes through API Management
3. App Service processes request and establishes SignalR connection
4. App Service retrieves agent configuration from Cosmos DB
5. Request is forwarded to Azure OpenAI Service with streaming enabled
6. Tokens are streamed back through SignalR to client in real-time
7. If tool calls are needed, Azure Functions execute the appropriate tools
8. Authenticated actions use the user's delegated token to make API calls
9. Metrics are logged to Application Insights throughout the process

Agent Configuration & Deployment Flow

1. Developer creates/updates agent configuration in development environment
2. Changes are committed to Azure DevOps repository
3. CI/CD pipeline runs offline evaluations against test cases
4. Results are stored and an approval request is created
5. Approvers review changes and evaluation results
6. Upon approval, configuration is stored in Cosmos DB with new version
7. Feature flag is created for gradual rollout (A/B testing)
8. Online evaluations monitor performance in production
9. Based on metrics, rollout percentage is gradually increased

Key Technical Components

Streaming Implementation

Azure SignalR Service provides WebSocket connections for token-by-token streaming, while Azure OpenAI Service supports streaming API responses.

Configuration System

Azure Cosmos DB stores versioned agent configurations including base model, prompts, tool configurations, and parameters.

Authentication Actions

Azure AD provides delegated tokens allowing the agent to perform authenticated actions on behalf of users via OAuth 2.0 flows.

Evaluation Framework

Combines offline evaluations (pre-deployment) and online metrics (post-deployment) with Azure DevOps for version control and bisecting issues.